

## COMBINATORIAL FITNESS FUNCTION CIRCUIT

### 1 TECHNICAL FIELD OF THE INVENTION:

2 The present invention relates generally to a genetic algorithm machine for  
3 executing genetic algorithms, and more specifically to an apparatus for computing the  
4 fitness of a chromosome that represents a trial solution to a combinatorial-type  
5 optimization problem to be solved by the genetic algorithm machine.

### 6 BACKGROUND:

7 Although evolutionary computing has roots as far back as the 1950s, genetic  
8 algorithms (hereinafter referred to by the initials GA) were introduced in 1975 by John  
9 Holland as a method for finding an optimum or near optimum solution to complicated  
10 problems. As noted by another researcher, Grefenstette, the GA is a useful method for  
11 finding optimum solutions to the Traveling Salesman Problem, a classic and well-known  
12 computationally intractable problem.

13 With reference now to FIGURE 1, there is illustrated therein a conceptual model  
14 of evolution in a genetic algorithm and how a solution to a problem evolves in processing  
15 the GA, generally designated by the reference numeral 100. As is understood in this art,  
16 in a genetic algorithm, an emulated chromosomal data structure is initially designed to  
17 represent a candidate or trial solution. A number of n-bit chromosomes of that data  
18 structure are then randomly generated and are registered in groups or populations of  
19 solutions. Parent chromosomes are selected from this population of generated  
20 chromosomes according to a given algorithm, *e.g.*, selected chromosomes 105 and 110 in  
21 FIGURE 1. Each generated chromosome is assigned a unique problem-specific fitness  
22 which may or may not differ from other chromosomes in the population, identifying the  
23 solution quality of the chromosome. The problem-specific fitness is expressed by a  
24 fitness value, as is known in the art. In a true evolutionary, survival of the fittest manner,  
25 particular chromosomes are selected from the population of chromosomes in proportion  
26 to their fitness values with more-fit chromosomes having a higher probability of being  
27 selected.

28 As further illustrated in FIGURE 1, when a pair of parent chromosomes, *e.g.*,  
29 chromosomes 105 and 110, are selected from the population, the parent chromosomes are  
30 combined using a probabilistically generated cut point, designated by the reference  
31 numeral 120. In the case of having no cutpoint generated, either of the parent  
32 chromosomes is simply copied to provide a new chromosome as a child chromosome.

1 Thus, a child chromosome is created and outputted. The child chromosome, therefore,  
2 contains portions of each parent or the whole portion of a parent, *e.g.*, a child  
3 chromosome 125 contains portion 105A of parent chromosome 105 and portion 110B of  
4 parent chromosome 110, as illustrated in FIGURE 1. The child chromosome may then be  
5 mutated in a controlled manner, preferably having a low probability. In the evolutionary  
6 example illustrated in FIGURE 1, the mutation is performed through inversion of a bit  
7 130 in the child chromosome 125, *e.g.*, 0 to 1 or 1 to 0. A mutated child chromosome  
8 125' is then evaluated to be assigned its fitness value. An evaluated child chromosome  
9 along with its fitness value is then stored as a member of the next generation in the  
10 population, perhaps replacing one or both of the associated parent chromosomes 105 and  
11 110.

12 After repeated iteration of this evolutionary process, the general fitness of  
13 chromosomes in the population improves to the optimal solution. Thus, a solution to the  
14 problem emerges in the population, and is acquired with highly-fit chromosomes  
15 concentrated in the population.

16 In a conventional approach, a GA is emulated by software and the algorithm used  
17 for computing the fitness of a GA-based candidate solution to the combinatorial problem  
18 is also emulated by software. Due to such a software-based emulation on conventional  
19 computers, however, the execution speed of the algorithm for finding an optimum  
20 solution to the combinatorial problem is extremely slow. Indeed, a major drawback of  
21 conventional genetic algorithm machines is the slow execution speed of a GA when  
22 emulated by software on conventional, general-purpose computers.

23 A hardware-based implementation of a GA has been devised for offsetting the  
24 drawback but only with a limited success in its execution speed. For example, U.S.  
25 Patent No. 5,970,487 to Shackleford, et al. solved some of the drawbacks and  
26 disadvantages of prior art techniques, particularly speed of operation, by the utilization of  
27 a hardware-based framework for accelerated used of genetic algorithms. The advantages  
28 and usages of the Shackleford et al. invention, Shackleford being the sole inventor in the  
29 instant application, are fully described in U.S. Patent No. 5,970,487, which is  
30 incorporated by reference herein.

31 The Shackleford et al. invention as described is an efficient problem-solving  
32 machine that may evolve solutions to many different problems. A common problem that  
33 is generally solved using a genetic algorithm is a combinatorial problem, also called a  
34 routing or ordering problem. A particularly intractable combinatorial problem is a so-

1 called non-deterministic polynomial hard (NP-hard) problem, which may be solved  
2 through a resource selection from among many resources, by an applied form of a GA,  
3 minimizing the hardware architecture, of a logic circuit, for example, and generating a  
4 resultant optimum solution.

5 An example of an NP-hard combinatorial problem is the aforementioned  
6 Traveling Salesman Problem (or TSP), as is known in the art, which can be used to model  
7 many combinatorial, routing and ordering problems. The TSP seeks to find the shortest  
8 route between  $n$  cities, and while any solution which contains all  $n$  cities once and only  
9 once is valid, some solutions are better than others. A solution to this problem describes  
10 the order of travel between cities, which determines the distance of the route traveled, so  
11 the order of travel between cities having the shortest route is the best solution. As is well  
12 understood in the mathematical and computer algorithm arts, the TSP is an NP-hard  
13 combinatorial problem with  $n!$  potential solutions and  $(n-1)!$  unique solutions.

14 With reference now to FIGURE 2, there is illustrated an example solution and its  
15 fitness for an 8-city Traveling Salesman Problem. The solution to the TSP is in the form  
16 of the order traveled and yields a fitness value calculated from a distance  $D_{x,y}$  of each new  
17 city from the last. In this example, the possible solution to the 8 city TSP {0, 4, 2, 7, 5, 6,  
18 3, 1} corresponds to the fitness value  $D_{Total} = D_{0,4} + D_{4,2} + D_{2,7} + D_{7,5} + D_{5,6} + D_{6,3} + D_{3,1} +$   
19  $D_{1,0}$ , as is understood to those skilled in the art. As illustrated in FIGURE 2, other routes  
20 are possible and, alternatively, a possible solution can include information of order  
21 traveled from any city to any other, with no repetition of cities. For simplicity, not all  
22 possible routes are shown in FIGURE 2.

23 Because of the large number of possible solutions to a TSP, *e.g.*, a 32-city TSP  
24 has over  $2.5 \times 10^{35}$  solutions, heuristic and non-deterministic solving methods must be  
25 used to solve this type of problem. The TSP can be solved, therefore, through an optimal  
26 solution-finding approach that aims at attaining an optimal or near optimal solution  
27 through a screening process of candidate or trial solutions created through a GA, based  
28 upon a fitness evaluation of the candidate solutions. In this approach, more-fit candidate  
29 solutions are selected with less-fit candidate solutions screened out to concentrate highly-  
30 fit solutions or chromosomes and in the end to reach an optimal or near optimal solution.

31 The Shackleford et al. invention as described hereinabove achieves a significant  
32 increase in execution speed in its hardware implementation. The Shackleford et al.  
33 invention is a general purpose GA machine that can solve any problem by using the  
34 appropriate chromosome generation template and fitness function circuit. However,

1 general purpose machines are always less efficient than task-specific dedicated machines,  
2 and the GA machine is no exception.

3 There is, therefore, a present need to provide a problem-specific, combinatorial-  
4 type fitness function circuit which performs a high speed fitness evaluation of candidate  
5 solutions generated through a GA. What is needed is, accordingly, an invention to  
6 provide a hardware-based fitness function circuit which accelerates the execution speed  
7 of a GA. What is needed is an invention to provide a hardware-based fitness function  
8 circuit.

9 The present invention describes a method to provide a high-speed, hardware-  
10 based fitness function that will match the performance of the hardware-based  
11 implementation of the GA.

12 It is, accordingly, an aspect of the present invention to provide a problem-specific,  
13 combinatorial-type fitness function circuit which performs a high speed fitness evaluation  
14 of candidate solutions generated through a GA.

15 It is another aspect of the present invention to provide a fitness function circuit  
16 which accelerates the execution speed of a GA.

17 It is a further aspect of the present invention to provide a hardware-based fitness  
18 function circuit.

#### 19 **SUMMARY:**

20 According to an embodiment of the present invention, a fitness function circuit for  
21 an execution of a genetic algorithm (GA) inputs a chromosome having n bits and outputs  
22 a fitness of the chromosome. The fitness is calculated as a solution to a combinatorial-  
23 type problem.

24 The fitness function circuit is a hardware circuit for calculating the cost of the  
25 elements indicated by the chromosome inputted, and then calculating the fitness of the  
26 chromosome based upon a calculated cost of elements indicated. The fitness function  
27 may include a solution register, a plurality of data tables, and a carry-save-adder.

28 The fitness function circuit receives a chromosome and stores the chromosome in  
29 the solution register. Each two consecutive values of the chromosome from the register  
30 are used to query multiple, identical data tables. Thus, by repeating the data tables and  
31 accessing the data tables in parallel, the circuit operates in a minimum amount of time.

32 Further scope of applicability of the present invention will become apparent from  
33 the detailed description given hereinafter. However, it should be understood that the  
34 detailed description and specific examples, while indicating preferred embodiments of the

invention, are given by way of illustration only, since various changes and modifications within the spirit and scope of the invention will become apparent to those skilled in the art from this detailed description.

#### **DESCRIPTION OF THE DRAWINGS:**

The features, aspects, and advantages of the present invention will become better understood with regard to the following description, appended claims, and accompanying drawings where:

FIGURE 1 illustrates a conceptual diagram of evolution in a GA machine, such as employed in the system and methodology of the present invention;

FIGURE 2 illustrates a representative example of a solution of a Traveling Salesman Problem;

FIGURE 3 depicts a flowchart illustrating a flow of the GA according to the present invention;

FIGURE 4 depicts a diagram of a fitness function circuit for combinatorial problems such as the Traveling Salesman Problem according to the present invention;

FIGURE 5 depicts an possible layout of a data table for use in the fitness function circuit; and

FIGURE 6 depicts a more efficient layout of a data table for use in the fitness function circuit.

#### **DETAILED DESCRIPTION:**

The following detailed description is presented to enable any person skilled in the art to make and use the invention. For purposes of explanation, specific nomenclature is set forth to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that these specific details are not required to practice the invention. Descriptions of specific applications are provided only as representative examples. Various modifications to the preferred embodiments will be readily apparent to one skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. The present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest possible scope consistent with the principles and features disclosed herein.

The present invention is directed to a fitness function circuit for a combinatorial problem, such as the Traveling Salesman Problem, optimized for use on a genetic algorithm (GA) machine, such as that set forth in Shackleford et al. To assist in the

1 general explanation of the operation of the fitness function circuit of a GA machine,  
2 reference is now made to FIGURE 3, which shows a flowchart of a GA with parent  
3 chromosomes P1 and P2, a child chromosome C, a mutated child chromosome C', and a  
4 fitness value F, all described in more detail hereinbelow.

5 With reference now to FIGURE 3, there is illustrated the first step in the flowchart  
6 of the GA, which is to create (step 305) a population of randomly generated  
7 chromosomes, evaluate their respective fitness values and store the chromosomes and  
8 their respective fitness values in a population memory 310.

9 A parent chromosome (generally designated by the reference symbol P) is then  
10 randomly selected (step 320) from the population memory 310 and assigned the first  
11 parent chromosome P1. It should be understood that when a parent chromosome is newly  
12 selected, the parent chromosome that was previously assigned the first parent  
13 chromosome P1 is re-assigned the second parent chromosome P2. The newly-selected  
14 parent chromosome P then becomes the first parent chromosome P1.

15 A child chromosome C is then created (step 330) from the two parent  
16 chromosomes P1 and P2, respectively, through a crossover process, such as described  
17 above in connection with FIGURE 1. In other words, the crossover process is a single-  
18 point crossover, whereby the first and second parent chromosomes P1 and P2 are divided,  
19 each at the same bit location, and the data to the left of that location in the first parent  
20 chromosome P1 is used to form the left part of a child chromosome C and the data  
21 inclusive of the bit and to the right in the second parent chromosome P2 is used to form  
22 the right part of the child chromosome C.

23 In a mutation step 340, each bit in the child chromosome C, for example, the  
24 aforescribed bit 130 in FIGURE 1, is exposed to the possibility of mutation. After one  
25 or more bits within the child chromosome C are flipped (or changed using another  
26 mechanism of random-like mutation), the mutated child chromosome C' is stored. In a  
27 preferred embodiment of the present invention, the probability of mutation for each bit is  
28 on the order of 1 percent.

29 After mutation, an evaluation of the child chromosome C' is made by a fitness  
30 function (step 350). A preferred fitness function is a re-configurable circuit which  
31 evaluates the problem-specific fitness of a child chromosome, as is understood in the art.

32 Finally, the survival of the mutated child chromosome C' is determined (step 360)  
33 based upon the fitness value F of the child chromosome C' outputted from the fitness  
34 function 350. For example, the fitness value F of the child chromosome C' is compared

1 with the least-fit fitness value of the least-fit chromosome stored in the population  
2 memory 310. If the child chromosome C' is more fit, then the child chromosome C'  
3 replaces the less-fit chromosome in the population memory 310. If, however, the child  
4 chromosome C' is less fit, then the child chromosome C' is simply discarded.

5 The repetitions of the steps of this process, *i.e.*, 320 to 360, shown in double lines,  
6 improve the quality of candidate solutions toward an optimum solution.

7 It should be understood that after repeated iteration of this process, the general  
8 fitness of chromosomes in the population improves. Thus, a solution to the problem  
9 emerges in the population, and an optimum or near optimum solution to the problem is  
10 acquired with highly-fit chromosomes becoming concentrated in the population.

11 Because of the iterative nature of the GA process, speed of execution is important.  
12 Each step 320 through 360 as described above is performed ideally during one clock  
13 cycle. The parent selection step 320, crossover step 330, mutation step 340, and survival  
14 determination step 360 are easily implemented in one clock cycle through hardware  
15 circuitry. The evaluation of each chromosome by the fitness function circuit 350 is the  
16 step that determines the speed of the execution of the problem solved by the GA machine.

17 When the GA machine is used to solve the TSP, as described hereinabove, the  
18 fitness function can be designed as a combinatorial problem evaluator. The measure of  
19 merit of a trial solution of an ordering or routing problem such as the TSP is the total  
20 distance of the route with a shorter distance having a higher fitness. The order of each  
21 solution is the important information in the solution, and the total distance of the route is  
22 directly related to the order of the solution. The fitness function circuit can be designed  
23 to use a table with  $n(n-1)/2$  entries that describe the distance from every city to every  
24 other city. According to each part of the solution being evaluated, the table would return  
25 a distance value, and for each part of the solution being evaluated, the distance returned  
26 would be summed into a total distance of the solution. However, to evaluate a potential  
27 solution to the TSP by serially accessing a single distance table would still require  $n$   
28 accesses to the table. The GA machine is capable of generating one trial solution per  
29 machine cycle, so a fitness function that can evaluate one solution per machine cycle is  
30 preferred.

31 With reference now to FIGURE 4 of the Drawings, there is illustrated therein a  
32 schematic drawing of a fitness function circuit, generally designated by the reference  
33 numeral 400, for use in combinatorial problems such as the TSP, as described  
34 hereinabove. Additionally, the fitness function preferably executes in one clock cycle.

1 As shown in FIGURE 4, the circuit includes a trial solution register 410 having  
2 component parts 411-418 therein, an array of  $n$  distance table RAMs 421-428  
3 (collectively designated by the reference numeral 420), and a carry-save-adder 430.

4 In operation, the circuit 400 receives a trial solution as input into the register 410  
5 either from the initial population memory 310 or from the mutation module 340, as  
6 described in the flowchart of a GA machine in FIGURE 3, and evaluates the fitness value  
7 of the solution. The fitness value of the solution is composed of the total of each part of  
8 the individual distances associated with each part of the chromosome, in the case of the  
9 TSP, or more generally, the individual values associated with the parts of the  
10 chromosome. The distances or values associated with every possible combination of  
11 chromosome are stored in the respective distance table RAM 420, which are each  
12 correlated to and accessed according to the order of each part of the values of the  
13 chromosome, *i.e.*, the individual bits within the trial solution register 410. The distances  
14 or values obtained from each data table RAM 420 are added in a carry-save adder 430  
15 and the total  $D_{Total}$  is outputted as the fitness value of the trial solution or chromosome.

16 The logical operation of the combinatorial fitness function circuit will be  
17 described in detail hereinbelow.

18 With reference again to the circuit shown in FIGURE 4, the fitness function  
19 circuit 400 receives a trial solution whose fitness is to be evaluated and stores that  
20 solution in the solution register 410. The solution register 410 is formed of multiple  
21 component parts 411-418. It should be understood that whether the trial solution is  
22 received from the initial population memory 310 or from the mutation module 340, the  
23 operation of the fitness function circuit 400 is unchanged. The solution register 410, of  
24 length eight values in the embodiment illustrated in FIGURE 4, is connected to each  
25 distance table RAM 421-428 in the array of distance table RAMs 420 in a sequential  
26 order from right to left.

27 Each distance table RAM in the array of distance table RAMs 420 receives an  
28 input signal from two separate values from the solution register 410. In particular, the  
29 input to each distance table RAM 421-428 in the array of distance table RAMs 420  
30 corresponds to a value from the distance table stored in the RAM, and the value retrieved  
31 from the table is transmitted as output.

32 As illustrated in FIGURE 4, the first value 411 of the register 410, in this case a  
33 three bit length value, is connected to the first distance table RAM 421 and the last  
34 distance table RAM 428. The second value 412 of the register 410 is connected to the



second distance table RAM 422 and the third distance table RAM 423, and so on. Likewise, the last value 418 of the register 410 is connected to the last distance table RAM 428 and the first distance table RAM 421.

With reference now to FIGURES 5 and 6, there are illustrated possible layouts of the distance table 420. As illustrated in FIGURE 5, a distance table, generally designated by the reference numeral 520, includes each possible value that can be received from the register 410 of FIGURE 4, in this case all the values between 0 and 7, in a grid. This table, however, is inefficient because it contains extra values. Because each value must be unique, the values in the distance table 520 representing the distance from one point to the same point are invalid, as indicated in the distance table 520 grid in FIGURE 5. The table also includes duplications, *i.e.*, the distance from one point to a second point is the same as the distance from the second point back to the first. A smaller table, therefore, containing only half the information of the table illustrated in FIGURE 5 is sufficient.

With reference now to FIGURE 6, there is illustrated an improved, abbreviated distance table, generally designated by the reference numeral 620. As shown, no values are repeated and the table 620 contains no invalid values of the distance between one point and that same point. Accordingly, table 620 is smaller and uses less memory, while containing the same information. Rather than being size  $(n)(n)$  like the table shown in FIGURE 5, the table shown in FIGURE 6 is size  $(n)(n-1)/2$ . The smaller size table is advantageous because it saves space in the circuit and memory on the RAM chip. It should be apparent to those skilled in the art that circuitry (not shown for simplicity) is included at the address input of the memory to insure that  $x > y$  for each  $D_{x,y}$  and that the decodable memory blocks are of such a granularity that they can be individually addressed as shown in FIGURE 6. It should be further understood that due to the limited capacity of current RAM chips, problems of data fragmentation or smearing may occur requiring the use of individual decodable blocks of RAM to provide the requisite address space, as is understood in the art.

With reference again to FIGURE 4, the value of the distance of each value of the solution to the next value of the solution is sent from each distance table RAM 421-428 in the array of distance table RAMs 420 to the aforementioned carry-save-adder 430. The total value  $D_{Total}$  from the carry-save-adder 430 is the fitness value of the trial solution.

The operation of the fitness function circuit is illustrated now with the example of FIGURE 2. In this case, the possible solution  $\{0, 4, 2, 7, 5, 6, 3, 1\}$  contains eight values representing eight cities with each city value having a bit length of three. This possible

1 solution is entered into the trial solution register 410 of FIGURE 4. The possible solution  
2 is then used to retrieve distance values from each distance table RAM 421-428 in the  
3 array of distance table RAMs 420.

4 Each value of the possible solution in the trial solution register 410 is used as one  
5 half of the address of the distance determined by the solution and retrieved from each  
6 distance table RAM 421-428 in the array of distance table RAMs 420. The first value  
7 411 of the possible solution, *i.e.*, {0}, when the solution is entered into the solution  
8 register 410 from right to left, is used as one part of the address transmitted to the first  
9 distance table RAM 421 and one part of the address transmitted to the second distance  
10 table RAM 422, as illustrated in FIGURE 4. Similarly, the second value 412 of the  
11 possible solution, {4}, is used as one part of the address transmitted to the second  
12 distance table RAM 422 and one part of the address transmitted to the third distance table  
13 RAM 423. Likewise, for the intermediate possible solution values, *i.e.*, 413-417, {2, 7, 5,  
14 6, 3}, and distance table RAMs 424-427. Finally, the last value 418 of the possible  
15 solution, {1}, is used as one part of the address transmitted to the last distance table RAM  
16 428 and one part of the address transmitted to the first distance table RAM 421, as  
17 illustrated in FIGURE 4. Thus, the respective addresses transmitted to each distance table  
18 RAM 421-428 in the array of distance table RAMs 420 are {04} to the first distance table  
19 RAM 421, {42} to the second distance table RAM 422, {27} to the third distance table  
20 RAM 423, and so on to the address transmitted to the last distance table RAM 328 being  
21 {10}. In this way, each distance table RAM 421-428 in the array of distance table RAMs  
22 420 receives an address of a specific order, each address comprised of two sequential  
23 values of the possible solution.

24 Each address transmitted to each distance table RAM in the array of distance table  
25 RAMs 420 retrieves a distance value related to address. With reference to the possible  
26 distance table layout of FIGURE 5, each part of the address corresponds to a value on the  
27 rows of the table and on the columns. For example, the address {42} transmitted to the  
28 second distance table RAM 422, corresponds to the value  $D_{4,2}$  at row 4 and column 2.  
29 The value  $D_{4,2}$  corresponds to the distance between city-4 and city-2. It should therefore  
30 be understood that each value on the table corresponds to the appropriate distance.

31 With reference again to FIGURE 4, the values retrieved from each distance table  
32 RAM 421-428 in the array of distance table RAMs 420, according to the possible solution  
33 of FIGURE 2, are sent to the carry-save-adder 430 and added in parallel. Accordingly,  
34 the output  $D_{Total}$  is the total sum of the distances between each value of the possible

1 solution. In this case,  $D_{Total} = D_{0,4} + D_{4,2} + D_{2,7} + D_{7,5} + D_{5,6} + D_{6,3} + D_{3,1} + D_{1,0}$ , *i.e.*, the  
2 particular distance a traveler must make to visit all of the cities in that order.

3 It should be understood that the output  $D_{Total}$  can be used as a fitness value in the  
4 genetic algorithm. In general, the output  $D_{Total}$  is the total sum of the values related to  
5 each value of the possible solution. The output  $D_{Total}$  can, therefore, be employed as a  
6 fitness value in the genetic algorithm where a higher fitness is denoted by a higher fitness  
7 value, or in the case of the Traveling Salesman Problem, a lower distance value, and a  
8 lower fitness is denoted by a lower fitness value, or in the case of the Traveling Salesman  
9 Problem, a higher distance value.

10 This implementation of a combinatorial fitness function circuit, such as the  
11 embodiment of the present invention depicted in FIGURE 4, with multiple repeated,  
12 identical data tables, evaluates each solution in one clock cycle, as required. This  
13 implementation allows the GA machine to operate at its full potential. It should be  
14 further understood that the implementation of the present invention is also scalable, with  
15 no loss of throughput beyond the simple embodiment shown in FIGURE 4 and described  
16 hereinabove. For a larger combinatorial problem, such as, for instance, a 32-city TSP,  
17 then 32 distance tables are required. Although the larger problem uses more memory,  
18 each solution to the larger problem is evaluated in the same amount of time.

19 The foregoing description of the present invention provides illustration and  
20 description, but is not intended to be exhaustive or to limit the invention to the precise  
21 one disclosed. Modifications and variations are possible consistent with the above  
22 teachings or may be acquired from practice of the invention. Thus, it is noted that the  
23 scope of the invention is defined by the claims and their equivalents.